# Homework 4 Introduction to Big Data Systems

Christoffer Brevik

October 13, 2024

## 1 Introduction

This documents contains my answers for the questions in homework 4, based on the given PDF describing the Google File System.

## 2 Part 1

### Q1: How does the master node get the locations of each chunks at startup

At startup, the master node does not store the chunk location information persistently. Instead, it retrieves the location of each chunk by polling all chunkservers. Each chunkserver reports the chunks it holds, and the master node updates its information accordingly. Additionally, whenever a chunkserver joins the cluster, the master node updates the chunk locations. The master keeps this information updated by sending periodic *HeartBeat* messages, making sure it is updated on chunkplacement and the chunkserver status. This approach ensures that the master always has up-to-date information about chunk locations in the system.

### Q2: What is the benefit of this approach comparing with the approach that the master persists this information?

The main benefit of this approach is the reduction in complexity related to maintaining consistency between the master and the chunkservers. Persisting chunk location information would require the system to handle various events such as chunkserver failures, renaming, and rejoining, which can lead to stale or inconsistent information. By polling the chunkservers at startup and using regular HeartBeat messages, the master node avoids the issues of synchronization and ensures accurate, up-to-date chunk location information at all times. This also simplifies the system's design, making it more robust against common failures in a distributed environment.

## 3 Part 2

We assume a cluster of 1000 servers, each server having 10 disks with 10TB storage capacity and 100MB/s I/O bandwidth per disk. The servers are connected by a 1Gbps (125MBps) ethernet cables, as nothing else is written I assume this bandwidth is per machine and not the total transfer cap in the system.

### Q1: What is the minimum time required to recovery a node failure

The total I/O bandwidth of all disks in a node is:

$$\text{Total bandwidth} = 10\,\text{disks} \times 100\,\text{MB/s} = 1000\,\text{MB/s}$$

The network bandwidth available per node is:

$$\text{Network bandwidth} = 1\,\text{Gbps} = 1024\,\text{Mbps} = 128\,\text{MB/s}$$

Since the network bandwidth is lower than the total I/O bandwidth of the disks, the recovery time will be limited by the network speed. Furthermore, we assume that since we are calculating the minimum time requred, that all other 999 nodes will use their resources to assist upon a node failure. Assuming that the 999 remaining servers work in perfect parallel to recover the data from the failed node, the total network bandwidth available is:

$$\text{Total bandwidth} = 999 \times 128\,\text{MB/s} = 127872\,\text{MB/s}$$

To transfer 100 TB of data at this rate, the time required is:

$$\text{Time to recover} = \frac{100 \times 1024 \times 1024\,\text{MB}}{127872\,\text{MB/s}} = 820\,\text{s} = 13,667\text{m}$$

Thus, the minimum time required to recover a node failure, assuming all other servers participate in the recovery, is approximately 14 minutes.

## Q2: Time to recover a failure node with throttled recovery bandwidth

Since the recovery traffic is throttled to 100 Mbps per machine, roughly a tenth of the original 1Gbps, we would expect a recovery time ten times as long, as the recovery time is directly proportional to the network. Here we still assume that all other 999 nodes, as nothing else is stated in the assignment. We also assume all other requirements are similar. Below are my calculations:

$$\text{Total bandwidth} = 999 \times \frac{100\,\text{Mbps}}{8\,\text{MB/Mb}} = 999 \times 12.5\,\text{MB/s} = 12487.5\,\text{MB/s}$$

At this throttled rate, the time required to recover 100 TB of data is:

$$\text{Time to recover} = \frac{100 \times 1024 \times 1024\,\text{MB}}{12487.5\,\text{MB/s}} = 8397\,\text{seconds}$$

Converting this to hours:

$$\text{Time to recover} = \frac{8392\,\text{seconds}}{3600\,\text{seconds/h}} \approx 2.33\,\text{hours.}$$

Thus, the time required to recover a node failure when the recovery traffic is throttled is approximately 2 hours and 20 minutes.

## Q3: How many server failures are likely to occur in a year in this cluster? What is the mean time between node failures in this cluster?

We know that each server node has a mean time between failures (MTBF) of 10,000 hours.

$$\frac{24\,\text{h/day} \times 365\,\text{days/year}}{10,000\,\text{h MTBF}} = \frac{8,760\,\text{h/year}}{10,000\,\text{h/failure}} = 0.876\,\text{failures/year/server.}$$

As this failure rate is independent for each server node, we have to multiply this by the total amount of serves in the cluster to estimate the expected number of failures per year in the cluster:

$$1000\,\text{servers} \times 0.876\,\text{failures/year/server} = 876\,\text{failures/year.}$$

The mean time between failures (MTBF) for the entire cluster is therefore:

$$\frac{8,760\,\text{hours/year}}{876\,\text{failures/year}} = 10\,\text{hours.}$$

Thus, the cluster is expected to experience one server failure approximately every 10 hours.

## Q4: What is the implication of the number of replicas used in GFS based on the results from Q2 and Q3?

The comparison between the recovery time (Q2) and the mean time between node failures (Q3) highlights the critical importance of replication in GFS. With recovery taking approximately 2.33 hours when throttled (as calculated in Q2), and with the cluster experiencing a node failure approximately every 10 hours (from Q3), it's evident that replication is essential to prevent data loss. The default number of replicas in GFS is three, which ensures redundancy and availability of data even when nodes fail.

The chances of multiple replicas experiencing data loss due to node failues are slim, and the chances of this is greatly reduces for each copy. Still, increasing the number leads to a reduction in performance as more updates have to be completed for each chunk write. Still, the default of three replicas strikes a good balance for most use cases. It provides sufficient fault tolerance while keeping the storage overhead manageable. Increasing the number of replicas could provide additional safety in environments with higher failure rates, but this would come at the cost of additional storage requirements. Conversely, reducing the number of replicas would expose the system to an increased risk of data loss in the event of multiple simultaneous failures, which GFS is designed to avoid.